This listing of claims will replace all prior versions, and listings, of claims in the application.

Listing of Claims:

1. (Currently Amended) A system for index key normalization in a database comprising a processor adapted for:

(a) selecting ~~a column of~~ an index key for normalization, wherein the index key contains more than one column, and wherein each column of the index key has a type, and wherein the index key contains more than one type of column type;

~~(b) generating a marker corresponding to the selected column, wherein the marker acts as a header for a normalized column value;~~

(b) tracking column normalization, wherein tracking column normalization comprises:

creating a tracking variable, wherein the tracking variable has a maximum value determined by a number of columns in the index key;

incrementing the tracking variable value each time one of the columns of the index key is normalized; and

determining that normalization of the index key is complete when the maximum value of the tracking variable value is reached;

(c) generating ~~the~~ a normalized ~~column~~ value for the index key, wherein generating the normalized value for the index key comprises ~~corresponding to the selected column, wherein generating the normalized column value comprises~~:

determining a column type for a ~~of the~~ column; ~~value, wherein each column type has an associated transformative function~~[[,]] and

determining an associated transformative function relating to the type of the column; and

applying the associated transformative function to ~~the column~~ a value of the column resulting in the normalized column value; ~~and~~

(d) generating a marker corresponding to the column, wherein the marker acts as a header for the normalized column value;

(e) storing a normalized index key value, wherein the normalized index key value is updated for each column value that is normalized; and

[[(d)]] (f) appending the marker and the normalized column value to a previously generated marker and normalized column value.

2. (Original) The system of claim 1, wherein the processor repeats steps (a) – (d) for each column in the index key.

3. (Original) The system of claim 1, wherein the processor, after generating the marker, determines if a column value is null, determines if a column value is of type bit, and determines if the column is sorted in ascending or descending order.

4. (Original) The system of claim 3, wherein the processor modifies the marker to indicate if the column value is null, is of type bit, and if the column is sorted in ascending or descending order.

5. (Cancelled)

6. (Previously Presented) The system of claim 1, wherein the processor compares the normalized column value to other normalized column values independent of the original column type.

7. (Original) The system of claim 1, wherein the processor standardizes the size of the marker and the normalized column value pair before appending the marker and the normalized column value pair to the previously generated marker and normalized column value pair if any.

8. (Original) The system of claim 7, wherein the processor standardizes the size of the marker and the normalized column value pair by comparing the marker and normalized column value pair to a predetermined maximum size and computing a checksum using the pair if the pair is greater than the predetermined maximum size; truncating the marker and

normalized column value pair by removing bits from the end of the pair in excess of the predetermined maximum size; and replacing the end bytes of the truncated pair with the computed checksum.

9. (Currently Amended) A system for index key column unnormalization of a normalized index key comprising a processor adapted for:

for a normalized index key that includes normalization of multiple columns of multiple types, selecting a column;

determining ~~if~~ that ~~the~~ a type of [[a]] the selected column ~~value~~ can be unnormalized; ~~and~~

~~if so~~[[,]] determining ~~if~~ that the selected column was not truncated; and

generating ~~the~~ an unnormalized column value, wherein generating the unnormalized column value comprises applying a reverse of a transformative function that is associated with the selected column ~~if the selected column was not truncated~~.

10. (Original) The system of claim 9, wherein the processor moves through the normalized index key until the selected column is reached, by:

(a) determining if the current column is a fixed size or variable size type; and

(b) if the current column is a fixed size type, moving forward in the normalized index key a number of bytes equal to the size of the fixed size type, or if the current column is a variable size type, moving forward in the normalized index key a number of bytes equal to the length of the column, determined by examining each byte of the current column until the end of the column is reached.

11. (Original) The system of claim 10, wherein the processor repeats steps (a) and (b) for each column in the normalized index key until the selected column is reached.

12. (Original) The system of claim 9, wherein the processor determines if the selected column was truncated by determining if the selected column is a fixed size or variable size type; and, if the selected column is a fixed size type, determining if the size of the previous columns in the index key plus the size of the selected column is greater than a

predetermined maximum size, or, if the selected column is a variable size type, examining each byte of the selected column until the end of the column is reached or the number of bytes examined plus the size of the previous columns in the index key exceed a predetermined maximum size.

13. (Currently Amended) A computer-readable medium having stored thereon computer-executable instructions for performing a method for index key normalization in a database comprising:

(a) selecting ~~a column of~~ an index key for normalization, wherein the index key contains more than one column, and wherein each column of the index key has a type, and wherein the index key contains more than one type of column type;

~~(b) generating a marker corresponding to the selected column, wherein the marker acts as a header for a normalized column value;~~

(b) tracking column normalization, wherein tracking column normalization comprises:

creating a tracking variable, wherein the tracking variable has a maximum value determined by a number of columns in the index key;

incrementing the tracking variable value each time one of the columns of the index key is normalized; and

determining that normalization of the index key is complete when the maximum value of the tracking variable value is reached;

(c) generating ~~the~~ a normalized ~~column~~ value for the index key, wherein generating the normalized value for the index key comprises ~~corresponding to the selected column, wherein generating the normalized column value comprises~~:

determining a column type for a ~~of the~~ column; ~~value, wherein each column type has an associated transformative function~~[[,]] and

determining an associated transformative function relating to the type of the column; and

applying the associated transformative function to ~~the column~~ a value of the column resulting in the normalized column value; ~~and~~

(d) generating a marker corresponding to the column, wherein the marker acts as a header for the normalized column value;

(e) storing a normalized index key value, wherein the normalized index key value is updated for each column value that is normalized; and

[[(d)]] (f) appending the marker and the normalized column value to a previously generated marker and normalized column value.

14. (Original) The computer-readable medium of claim 13, further comprising computer-executable instructions for repeating steps (a) – (d) for each column in the index key.

15. (Original) The computer-readable medium of claim 13, further comprising computer-executable instructions for, after generating the marker:

determining if a column value is null;

determining if a column value is of type bit; and

determining if the column is sorted in ascending or descending order.

16. (Original) The computer-readable medium of claim 15, further comprising computer-executable instructions for modifying the marker to indicate if the column value is null, is of type bit, and if the column is sorted in ascending or descending order.

17. (Cancelled)

18. (Previously Presented) The computer-readable medium of claim 13, further comprising computer-executable instructions for comparing the normalized column value to other normalized column values independent of the original column type.

19. (Original) The computer-readable medium of claim 13, further comprising computer-executable instructions for standardizing the size of the marker and the normalized

column value pair before appending the marker and the normalized column value pair to the previously generated marker and normalized column value pair if any.

20. (Original) The computer-readable medium of claim 19, wherein standardizing the size of the marker and the normalized column value pair comprises:

comparing the marker and normalized column value pair to a predetermined maximum size and computing a checksum using the pair if the pair is greater than the predetermined maximum size;

truncating the marker and normalized column value pair by removing bits from the end of the pair in excess of the predetermined maximum size; and

replacing the end bytes of the truncated pair with the computed checksum.

21. (Currently Amended) A computer-readable medium having stored thereon computer-executable instructions for performing a method for index key column unnormalization of a normalized index key comprising:

for a normalized index key that includes normalization of multiple columns of multiple types, selecting a column;

determining ~~if~~ that ~~the~~ a type of [[a]] the selected column ~~value~~ can be unnormalized; ~~and~~

~~if so~~[[,]] determining ~~if~~ that the selected column was not truncated; and

generating ~~the~~ an unnormalized column value, wherein generating the unnormalized column value comprises applying a reverse of a transformative function that is associated with the selected column ~~if the selected column was not truncated~~.

22. (Original) The computer-readable medium of claim 21, further comprising computer-executable instructions for moving through the normalized index key until the selected column is reached, by:

(a) determining if the current column is a fixed size or variable size type; and

(b) if the current column is a fixed size type, moving forward in the normalized index key a number of bytes equal to the size of the fixed size type, or if the current column is a variable size type, moving forward in the normalized index key a number of bytes equal to

the length of the column, determined by examining each byte of the current column until the end of the column is reached.

23. (Original) The computer-readable medium of claim 22, further comprising computer-executable instructions for repeating steps (a) and (b) for each column in the normalized index key until the selected column is reached.

24. (Original) The computer-readable medium of claim 21, wherein determining if the selected column was truncated comprises:

determining if the selected column is a fixed size or variable size type; and

if the selected column is a fixed size type, determining if the size of the previous columns in the index key plus the size of the selected column is greater than a predetermined maximum size, or, if the selected column is a variable size type, examining each byte of the selected column until the end of the column is reached or the number of bytes examined plus the size of the previous columns in the index key exceed a predetermined maximum size.

25. (Currently Amended) A method for index key normalization in a database comprising:

(a) selecting ~~a column of~~ an index key for normalization, wherein the index key contains more than one column, and wherein each column of the index key has a type, and wherein the index key contains more than one type of column type;

~~(b) generating a marker corresponding to the selected column, wherein the marker acts as a header for a normalized column value;~~

(b) tracking column normalization, wherein tracking column normalization comprises:

creating a tracking variable, wherein the tracking variable has a maximum value determined by a number of columns in the index key;

incrementing the tracking variable value each time one of the columns of the index key is normalized; and

determining that normalization of the index key is complete when the maximum value of the tracking variable value is reached;

(c) generating ~~the~~ a normalized ~~column~~ value for the index key, wherein generating the normalized value for the index key comprises ~~corresponding to the selected column, wherein generating the normalized column value comprises~~:

determining a column type for a ~~of the~~ column; ~~value, wherein each column type has an associated transformative function~~[[,]] and

determining an associated transformative function relating to the type of the column; and

applying the associated transformative function to ~~the column~~ a value of the column resulting in the normalized column value; ~~and~~

(d) generating a marker corresponding to the column, wherein the marker acts as a header for the normalized column value;

(e) storing a normalized index key value, wherein the normalized index key value is updated for each column value that is normalized; and

[[(d)]] (f) appending the marker and the normalized column value to a previously generated marker and normalized column value.

26. (Original) The method of claim 25, further comprising repeating steps (a) – (d) for each column in the index key.

27. (Original) The method of claim 25, further comprising, after generating the marker:

determining if a column value is null;

determining if a column value is of type bit; and

determining if the column is sorted in ascending or descending order.

28. (Original) The method of claim 27, further comprising modifying the marker to indicate if the column value is null, is of type bit, and if the column is sorted in ascending or descending order.

29. (Cancelled)

30. (Previously Presented) The method of claim 25, further comprising comparing the normalized column value to other normalized column values independent of the original column type.

31. (Original) The method of claim 25, further comprising standardizing the size of the marker and the normalized column value pair before appending the marker and the normalized column value pair to the previously generated marker and normalized column value pair if any.

32. (Original) The method of claim 31, wherein standardizing the size of the marker and the normalized column value pair comprises:

comparing the marker and normalized column value pair to a predetermined maximum size and computing a checksum using the pair if the pair is greater than the predetermined maximum size;

truncating the marker and normalized column value pair by removing bits from the end of the pair in excess of the predetermined maximum size; and

replacing the end bytes of the truncated pair with the computed checksum.

33. (Currently Amended) A method for index key column unnormalization of a normalized index key comprising:

for a normalized index key that includes normalization of multiple columns of multiple types, selecting a column;

determining ~~if~~ that ~~the~~ a type of [[a]] the selected column ~~value~~ can be unnormalized; ~~and~~

~~if so~~[[,]] determining ~~if~~ that the selected column was not truncated; and

generating ~~the~~ an unnormalized column value, wherein generating the unnormalized column value comprises applying a reverse of a transformative function that is associated with the selected column ~~if the selected column was not truncated~~.

34. (Original) The method of claim 33, further comprising moving through the normalized index key until the selected column is reached, by:

(a) determining if the current column is a fixed size or variable size type; and

(b) if the current column is a fixed size type, moving forward in the normalized index key a number of bytes equal to the size of the fixed size type, or if the current column is a variable size type, moving forward in the normalized index key a number of bytes equal to the length of the column, determined by examining each byte of the current column until the end of the column is reached.

35. (Original) The method of claim 34, further comprising repeating steps (a) and (b) for each column in the normalized index key until the selected column is reached.

36. (Original) The method of claim 33, wherein determining if the selected column was truncated comprises:

determining if the selected column is a fixed size or variable size type; and

if the selected column is a fixed size type, determining if the size of the previous columns in the index key plus the size of the selected column is greater than a predetermined maximum size, or, if the selected column is a variable size type, examining each byte of the selected column until the end of the column is reached or the number of bytes examined plus the size of the previous columns in the index key exceed a predetermined maximum size.

37. (New) A computer-readable storage medium having stored thereon computer-readable instructions that, when executed by a computer, cause the computer to perform a process normalizing index keys in a database, the process comprising:

(a) selecting an index key for normalization, wherein the index key contains more than one column, and wherein each column of the index key has a type, and wherein the index key contains more than one type of column type;

(b) tracking column normalization, wherein tracking column normalization comprises:

creating a tracking variable, wherein the tracking variable has a maximum value determined by a number of columns in the index key;

incrementing the tracking variable value each time one of the columns of the index key is normalized; and

determining that normalization of the index key is complete when the maximum value of the tracking variable value is reached;

(c) selecting a first column of the index key, wherein the first column of the index key has a first column value;

(d) generating a normalized value for the first column of the index key, wherein generating the normalized value for the first column comprises:

determining a column type of the first column, wherein each column type has an associated transformative function;

determining the associated transformative function based on the type of the first column; and

applying the associated transformative function to the value of the first column resulting in the normalized value of the first column; and

(e) storing a normalized index key value, wherein the normalized index key value is updated each time a column value is normalized;

(f) determining that a size of the normalized index key value is smaller than a predetermined normalized index key maximum size;

(g) incrementing the value of the tracking variable;

(h) selecting a second column of the index key, wherein the second column of the index key has a second column value;

(i) generating a normalized value of the second column of the index key, wherein generating the normalized value of the second column comprises:

determining a column type of the second column, wherein each column type has an associated transformative function;

determining the associated transformative function based on the type of the second column; and

applying the associated transformative function to the value of the second column resulting in the normalized value of the second column; and

(j) updating the normalized index key value to include the normalized value of the second column of the index key; and

(k) incrementing the value of the tracking variable;

(l) checking whether the value of the tracking variable has reached the maximum value.

38. (New) The computer readable storage medium of claim 37, wherein the index keys are part of a b-tree database.

39. (New) The computer readable storage medium of claim 38, further comprising compressing the b-tree database, wherein compressing the b-tree database comprises:

(a) selecting a first normalized index key of the b-tree database;

(b) leaving the first normalized index key uncompressed

(c) selecting a second normalized index key of the b-tree database; and

(d) compressing the b-tree database by storing only the difference between the first normalized index key and the second normalized index key.